Detektion von Markern in einer Szene

Bachelorarbeit

an der Fakultät für Informations-, Medien und Elektrotechnik der

Fachhochschule Köln im Institut für Medien- und Phototechnik

Autor

Felix Spalthoff

Matr.-Nr.: 11050440

Referent: Prof. Dr. rer. nat. Dietmar Kunz

Co-Referent: Prof. Dr.-Ing. Gregor Fischer

Köln, November 2008

Marker Detection in a Scene

Bachelorthesis

at the Faculty of Information, Media and Electrical Engineering at the Cologne University of Applied Sciences in the Institute of Media and Imaging Technology

Author

Felix Spalthoff

Matr.-No.: 11050440

First Reviewer: Prof. Dr. rer. nat. Dietmar Kunz

Second Reviewer: Prof. Dr.-Ing. Gregor Fischer

Cologne, November 2008

Kurzbeschreibung

Titel: Detektion von Markern in einer Szene

Autor: Felix Spalthoff

Referenten: Prof. Dr. rer. Nat. Dietmar Kunz
Prof. Dr.-Ing. Gregor Fischer

Zusammenfassung:

In dieser Arbeit wird ein Detektionsverfahren vorgestellt, das einfarbige, kreisförmige Objekte in einem digitalen Bild erkennt. Die Methode umfasst eine Farbsegmentierung, eine Berechnung des Distanzmaßes und eine Überprüfung der Form. Der Algorithmus ist in der Programmiersprache Java als Plugin für die Bildverarbeitungssoftware ImageJ geschrieben.

Stichwörter: Bildverarbeitung, Java, ImageJ, Detektion, Extraktion, Kreiserkennung

Sperrvermerk:

Kein Sperrvermerk

Datum: 04.11.2008

Abstract

Title: Marker Detection in a Scene

Author: Felix Spalthoff

Advisores: Prof. Dr. rer. Nat. Dietmar Kunz
Prof. Dr.-Ing. Gregor Fischer

Summary: This thesis provides a method to detect monochromatic circular-shaped objects in a digital image. The method consists of a color segmentation, a distance map calculation and a shape check. The algorithm is written in the programming language Java and runs as a plugin for the image processing software ImageJ.

Keywords: Image Processing, Java, ImageJ, Detection, Extraction, Circle recognition

Remark of Closure:

None

Date: 04.11.2008

Inhaltsverzeichnis

1 Einleitung

1.1 Zielsetzung der Arbeit

Im Zuge der automatischen Erfassung von Bildkomponenten und derer individuellen Weiterverarbeitung sind Algorithmen, die verlässlich Objekte in einem digitalen Bild erfassen unabdingbar. In vielen Bereichen haben Detektionsverfahren mittlerweile Einzug erhalten und sorgen für schnelle, automatisierte Prozesse.

Das Ziel dieser Arbeit ist es, eine wirksame Methode zu entwickeln, wie bestimmte Objekte (im Folgenden Marker genannt) in einem digitalen Bild detektiert werden können. Den Markern liegen hierbei festgelegte Kriterien zu Grunde:

- **Rundheit**

  Da die Marker in einem dreidimensionalen Abbild der Wirklichkeit erscheinen sollen, ist eine nahezu ideale Rundheit in allen Dimensionen vorteilhaft. In Realität ist dies zwar schwer zu erreichen, allerdings erfüllen Kugeln oder Bälle hierzu ihren Zweck.

- **bekannte Farbe**

  Damit sich die Marker vom Hintergrund abheben und so ihre Detektion erleichtern, haben sie eine festgelegte und einheitliche

Farbe. Gut geeignet ist eine auffälliger Farbton mit Signalcharakter, der eher selten in weiteren Bildelementen auftaucht. Im Folgenden ist dies ein Rot.

Aus diesen Eigenschaften sorgt ein in der Programmiersprache Java [1] geschriebenes Plugin für die Bildverarbeitungssoftware ImageJ [2] für eine Erkennung dieser Elemente in einem Bild. Als Entwicklungsumgebung für das Programm dient die Software-Plattform Eclipse [3]. Die Ausgabe der ermittelten Marker erfolgt eindeutig hinsichtlich Position und Größe in der digitalen Vorlage.

1.2 Einsatzgebiete der Markerdetektion

Ein großes Anwendungsgebiet von automatischen Detektionsverfahren liegt in der Medizin. Ob es sich um Röntgenbilder oder das bildgebende Verfahren der Magnetresonanztomographie handelt, die automatisierte Erkennung von Kreiselementen wird in vielen Bereichen angewandt [4, S. 7]. Insbesondere ist auch im mikrobiologischen Bereich eine hohe Genauigkeit gefordert, um beispielsweise bestimmte Mikroorganismen auffinden und behandeln zu können. Kreisförmige Bakterien, Pilze oder Protozoen[1] könnten bei eventueller Optimierung des Programms zur Markerdetektion erkannt werden.

Abbildung 1: Runde Ca-Protozoen in verschiedenen Stadien

Ein anderes Gebiet stellt die maschinelle Fertigung von Produkten dar. Generell kann ein Marker hierbei beispielhaft als Passmarke gesehen werden, die für ein korrektes Aneinandersetzen verschiedener Einzelteile sorgt. Als Anwendungsbeispiel können zweidimensionale Passmarken dienen, die im Druckwesen üblich sind.

[1] Einzeller, die keine Zellwand und im Gegensatz zu Bakterien einen Zellkern besitzen

Diese Markierungen sorgen für die richtige Ausrichtung von Druckplatten beim Mehrfarbdruck. Die Passmarken an sich können verschiedene Formen annehmen, üblich sind Passkreuze die meist Kreiselemente enthalten [2, S. 249].

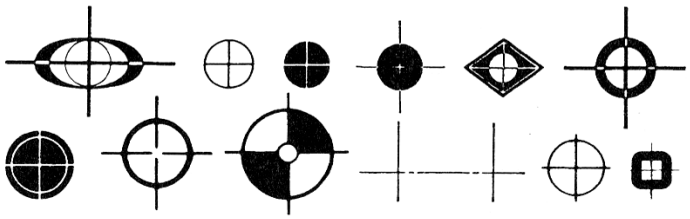


Abbildung 2: Verschiedene Passkreuze

Erkennung von bestimmten Bildelementen verbreitet sich auch immer mehr in der Digitalfotografie vor und während der Aufnahme eines Fotos. In verstärktem Maße entwickeln und verfeinern Kamerahersteller ihre Algorithmen zur Gesichts- und Objekterkennung in digitalen Kompakt- und auch Spiegelreflexkameras. Kreisförmige Objekte stehen hierbei zwar nicht direkt im Mittelpunkt, sind bei einer Detektion aber auch erfassbar. Die Kamerahersteller geben wenig über die genaue Art der Erkennung preis, meist handelt es sich aber um eine bildverarbeitungstechnische Analyse der Szene innerhalb des Fokusrahmens [12].

Eine über das Verfahren der bloßen Detektion hinausgehende Methode könnte den örtlichen Zusammenhang mehrerer Marker in einer Szene berücksichtigen. Abhängig davon, wie viele Marker sich in dem Bild befinden, kann durch die jeweilige Eigenschaft der Größe und Lage eines Markers eine perspektivische Definition des dargestellten Raums ermöglicht werden. Sind alle Marker gleich groß, zeichnen sich tiefer im Raum liegende im Bild durch einen kleineren Radius aus. Mit der Betrachtung aller drei Dimensionen könnte durch weiter entfernt liegende Marker ein Fluchtpunkt bestimmt werden, der für eine Rekonstruktion des Raums sorgt. Eine geeignete, im Bildkontext bewusst gewählte Position des Markers ist dabei vorausgesetzt.

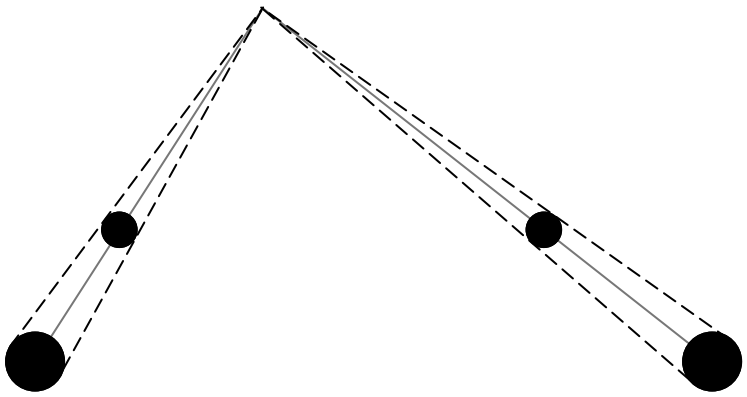


Abbildung 3: Skizzierung einer räumlichen Betrachtungsweise der Marker

2 Theoretische Grundlagen

2.1 Mathematische Verfahren

Die hier entwickelte Methodik zur „Detektion von Markern in einer Szene“ beruht auf verschiedenen mathematischen Berechnungen. Nachfolgend sind allgemein gültige Operationen und Operatoren aufgeführt, die in dem programmiertechnischen Teil Anwendung finden. Berechnungen und individuelle Methoden, die sich direkt auf den Programmablauf beziehen, werden in dem anschließenden Kapitel aufgegriffen.

2.1.1 Segmentierung

Die Segmentierung steht an der Grenze zwischen der ersten Stufe der Bildverarbeitung und der Bildanalyse [7, S. 471]. Der Ansatz liegt darin, bei dem Ursprungsbild geeignete lokale Objektmerkmale zu extrahieren und diese in einem neuen Binärbild aufzuzeigen. Die Merkmale lassen sich vielfältig wählen und generell in Kategorien wie pixelbasierte oder kantenbasierte Methoden und regionenbasierte Verfahren einteilen [7, S. 471 ff.]. Das letztendliche Ergebnis einer Segmentierung ist in der Regel ein Binärbild. Dieses entsteht auf die Weise, dass für jeden einzelnen Bildpunkt nach dem zuvor gewählten Kriterium eine Entscheidung getroffen wird, ob

dieser Bildpunkt zu dem Objekt gehört und im Binärbild den Wert 1 oder andernfalls 0 erhält.

Bei Objekten bekannter Farbe ist es sinnvoll eine pixelbasierte Segmentierung anhand der RGB-Werte vorzunehmen, um so ein Binärbild zu erhalten, das alle Elemente des vorliegenden Bildes mit genau dieser Farbe darstellt. Die Bildinformationen sind damit auf das Wesentliche reduziert und die Diskontinuitäten an den Rändern der Objekte lassen in dem schwarz-weißen Endergebnis Regionen erkennen.

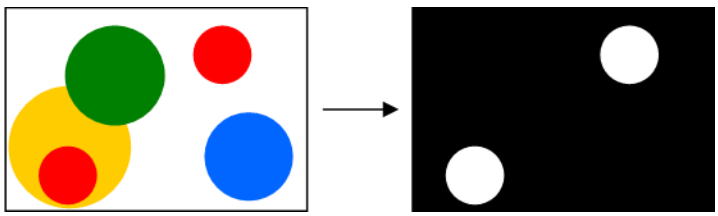


Abbildung 4: Beispiel einer Segmentierung anhand eines roten Farbwerts

Ob das Ergebnisbild schwarze Objekte vor weißem Hintergrund oder weiße Elemente vor schwarzer Fläche zeigt ist weitgehend unerheblich. Ein Informationsverlust ist dadurch nicht gegeben.

2.1.2 Median und Medianfilter

Der Median, auch Zentralwert genannt, bezeichnet den mittleren aller der Größe nach sortierten Variablenwerte. Liegt eine gegebene Menge an

Werten vor, so werden diese im ersten Schritt der Größe nach aufgelistet. Der Median kennzeichnet dann die Grenze, bei der gleich viele Werte vorhanden sind, die auf der einen Seite kleiner und auf der anderen Seite größer sind. Im Idealfall, bei einer ungeraden Anzahl an Werten, liegt der Median auf einem konkreten Wert, bei einer geraden Anzahl kann er aber auch dazwischen liegen.

Der Median erweist sich bei nicht-normalverteilten Verteilungsfunktionen als sinnvoll und ist resistent gegenüber ausreißenden Werten. Ein weiterer Vorteil ist, dass anders als bei dem arithmetischen Mittel keine neuen Werte erzeugt werden.

Der Medianfilter ist ein nichtlinearer Signalverarbeitungsfilter, der von John Wilder Tukey entwickelt wurde [5, S. 277]. Vorstellen lässt er sich als Filterkern, der pixelweise über das Bild fährt. Ausgangspunkt ist ein Grauwert- oder Binärbild. Die Berechnungsvorschrift ist folgende: Liste alle Pixelwerte, die sich im Bereich des Filterkerns befinden, der Größe nach auf und schreibe den Zentralwert in das Pixel, das sich in der Mitte des Kerns befindet.

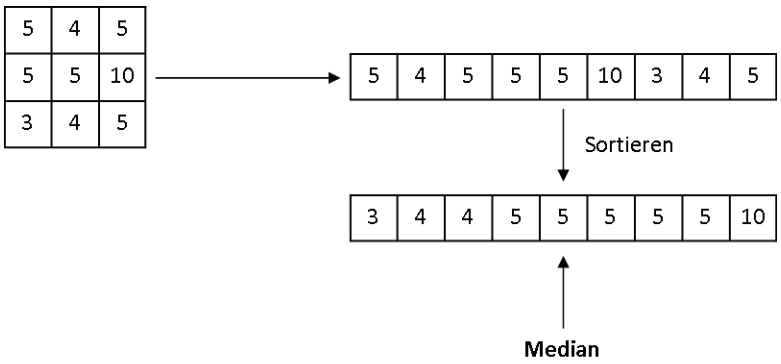


Abbildung 5: Vorgehensweise des Medianfilters am Beispiel eines Grauwertbildes

Der Medianfilter sorgt dafür, dass Pixel-Ausreißer eliminiert werden. Weicht ein Bildpunkt in seiner Helligkeit stark von seinen Umgebungspixeln ab, so wird dieses durch die Eigenschaft des Medians bei geeigneter örtlicher Begrenzung durch die Ausmaße des Filterkerns seinen Nachbarpixeln angeglichen.

Die Größe des Kerns, und somit auch der Grad der entstehenden Glättung, lässt sich dabei frei wählen und dadurch vorab grob auf den Bildinhalt abstimmen.

2.1.3 Distanzmaß

Das Distanzmaß, auch als Distanztransformation bezeichnet, ist ein nützliches Verfahren, um Entfernungen zu messen. In einem Binärbild wird für jeden Bildpunkt die Entfernung in Pixeln zur nächsten Kante bestimmt. Dieser Wert wird dann für jedes aktuell betrachtete Pixel an gleicher Stelle in ein neues Grauwertbild geschrieben [5, S. 626].

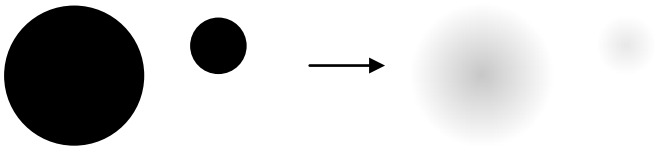


Abbildung 6: Ursprungsbild und Ergebnis nach Anwendung des euklidischen Distanzmaßes

Durch den iterativen Gedanken, für jedes Pixel eine Entfernung zu bestimmen, ist die Methode sehr aufwändig [7, S. 541]. Die Ergebnisse in dem Grauwertbild sind allerdings mitunter sehr aufschlussreich. In der Bildverarbeitung stellt das Distanzmaß ein nichtlineares, morphologisches Filter dar.

Die Distanztransformation kann unterschiedliche Abstände liefern. Abhängig von der betrachteten Pixelnachbarschaft kann das Resultat die Block- oder Schachbrettdistanz sein. Für Kreiselemente bietet sich aber die euklidische Distanz an, da sie morphologische Operationen isotrop macht. Für die Berechnung des euklidischen Abstands sind schnelle Algorithmen verfügbar [7, S 541].

2.2 Bekannte Verfahren zur Kreisdetektion

2.2.1 Kreis-Hough-Transformation

Die von Paul Hough entwickelte Methode, die gemeinhin als „Hough-Transformation“ bezeichnet wird, ist ein allgemeiner Ansatz, um beliebige, parametrisierbare Formen in Punktverteilungen zu finden [6, S. 156].

Speziell auf Kreiselemente bezogen ergeben sich drei Freiheitsgerade, die sich z. B. in folgende Form fassen lassen:

$$Circle = (\bar{x}, \bar{y}, \rho)$$

Ein Punkt $p = (u, v)$ liegt dabei auf einem Kreis, wenn die folgende Bedingung gilt [6, S. 167]:

$$(u - \bar{x})^2 + (v - \bar{y})^2 = \rho^2$$

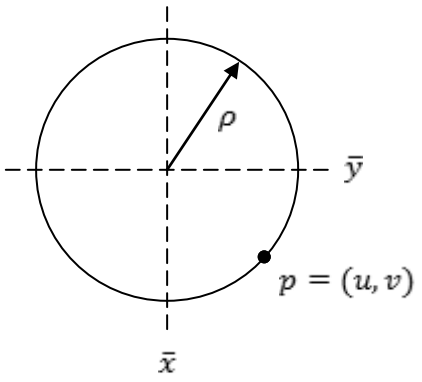


Abbildung 7: Parametrisierung von Kreisen

Die Parameter sind nun klar definiert. Um mit der Kreisdetektion zu beginnen, ist zuerst ein Kantenbild des Ausgangsbildes notwendig. Dieser Schritt ist vollkommen unabhängig zu der Hough-Transformation und das angewandte Verfahren zur Kantenbestimmung kann ein beliebiges sein [10, S. 3]. An jedem Kantenpunkt wird nun ein Kreis gezeichnet, der über einen zuvor bestimmten Radius verfügt. Der Mittelpunkt eines Kreises im Originalbild mit genau diesem Radius zeigt sich als der Punkt, an dem sich die meisten Kantenpunkt-Kreise schneiden [10, S. 3]. Es entsteht dort ein Maximalwert.

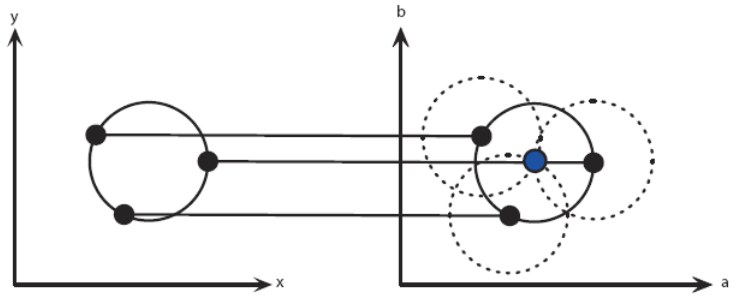


Abbildung 8: Kreis-Hough-Transformation für einen festgelegten Radius

Das Verfahren der Kreis-Hough-Transformation (CHT) ist sehr gebräuchlich, allerdings auch extrem rechenaufwändig. Für unbekannte Kreisradien, die im Ursprungsbild gesucht werden, muss das Verfahren iterativ für alle möglichen Kantenpunkt-Kreisradien durchlaufen werden [1, S. 1].

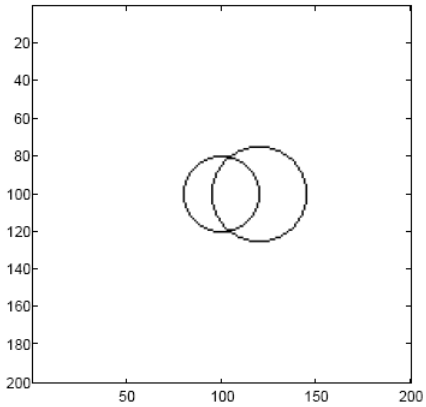


Abbildung 9: Anwendungsbeispiel für eine CHT;
Originalbild mit zwei Kreisen der Radien r1 = 20 und r2 = 25

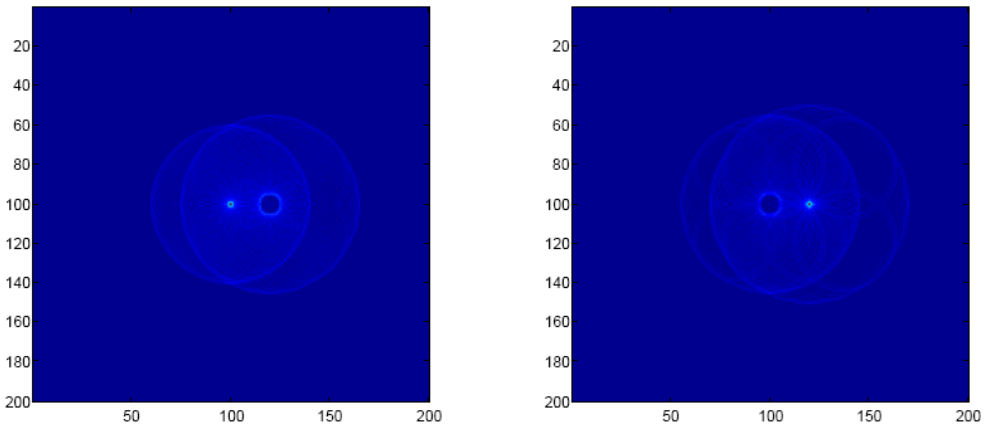


Abbildung 10: Ergebnis der CHT;
CHT mit r = 20 (links) und mit r = 25 (rechts)

2.2.2 Gradientenverfahren Fast Circle Detection

Das Verfahren mit der vollständigen Bezeichnung „Fast Circle Detection Using Gradient Pair Vectors“ (FCD) wurde 2003 auf der DICTA[2] von den iranischen Ingenieuren A. A. Rad, K. Faez und N. Qaragozlou vorgestellt [4]. Der Algorithmus versteht sich als Alternative zu der Kreis-Hough-Transformation (CHT) und soll Kreise, die sich in ihrer Helligkeit stark von dem Hintergrund unterscheiden sehr schnell und äußerst exakt erfassen können.

Das Verfahren beginnt mit einer Gradientenbestimmung. Angenommen es befindet sich ein dunkler Kreis vor hellem Hintergrund, so zeigen die Gradientenvektoren nach außen (Abb. 11).

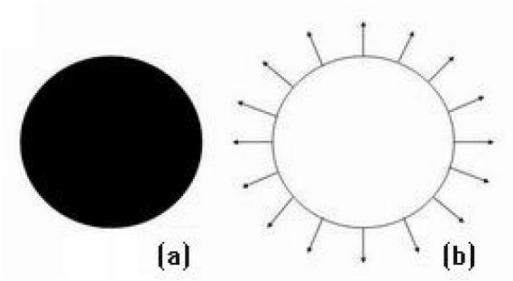


Abbildung 11: (a) Schwarzer Kreis vor weißem Hintergrund, (b) Gradientenvektoren von (a)

Aufgrund der Symmetrie ergibt sich, dass es für jeden Vektor an der gegenüberliegenden Seite des Kreises einen zugehörigen Vektor gibt. So

[2] DICTA ist die Abkürzung für *Digital Image Computing: Techniques and Applications* und ist eine internationale Konferenz zum Thema Bildverarbeitung

entsteht ein Vektorpaar, das über einen Winkel α von (zumindest nahezu) 180° verfügt. Als Folgerung daraus ergibt sich, dass der Winkel β nicht groß von 0° abweichen kann (Abb. 12).

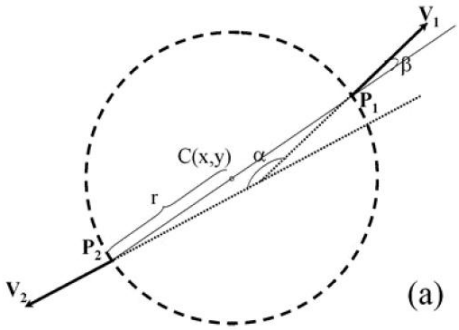


Abbildung 12: (a) Vektorpaare (V_1 & V_2) an einem Kreis

Nun werden alle Vektorpaare auf diese Art bestimmt. Vektoren, die in ihrer Orientierung exakt gegensätzlich sind, sich am Kreis aber nicht genau gegenüber liegen, werden durch die Winkelbedingung ausgeschlossen.

Der letzte Schritt umfasst die Bestimmung der wirklichen Kreise und das Ausschließen von überflüssigen Kreiskandidaten. Die Entwickler des Algorithmus zeigen neben dem Ansatz der CHT eine weitere, einfachere Herangehensweise auf, die in ihrer Methode Anwendung findet. Hierbei werden zuerst die Kreiskandidaten mit ihren Koordinaten und dem Radius als Tripel (C_x, C_y, r) gespeichert und anschließend unter Berücksichtigung der euklidischen Distanz betrachtet. Kreise können dann durch ein weiterführendes Verfahren mittels „seeds of clusters“ und Vektor- und Varianzberechnungen detektiert werden [1, S. 4].

Die Methode der FCD zeigt sehr schnelle Berechnungen und ist weniger speicherintensiv als andere Verfahren. Darüberhinaus kann durch eine bekannte Anzahl oder (bzw. und) Größe der Kreise die Performance des Algorithmus noch einmal erheblich verbessert werden [1, S. 4].

Das Ziel der Entwickler war es, den ihrer Ansicht nach größten Nachteil der CHT mit der FCD zu beseitigen. Dem enormen Rechenaufwand der CHT steht die FCD mit einer deutlich kürzeren Laufzeit gegenüber. Abhängig von der Größe des untersuchten Bildes kann diese 700- bis über 1000-mal schneller sein. Auch im Vergleich zu der Edge Oriented Circle Hough Transformation (EOCHT) ergeben sich klare Vorzüge (Abb. 13).

Image size 256×256	Original	Binary	Salt&Pepper	Gaussian
CHT	205	172	319	262
EOCHT	23	19.7	29	27
FCD	0.30	0.26	0.32	0.33
Image size 512×512	Original	Binary	Salt&Pepper	Gaussian
CHT	1711	1567	2073	1932
EOCHT	139	113	157	148
FCD	1.58	1.44	1.63	1.60

Abbildung 13: Ergebnisse der Laufzeiten verschiedener Kreisdetektionsverfahren (Zeitangaben in Sekunden)

Als einen weiteren Vorteil der FCD ergibt sich, dass dieser Algorithmus weniger anfällig gegenüber Pfeffer-und-Salz-Rauschen ist. Trägt man die Intensität des Rauschens gegenüber der Fehlerrate der Detektion auf, so ergibt sich ein deutlich schlechteres Verhalten für die CHT (Abb. 14).

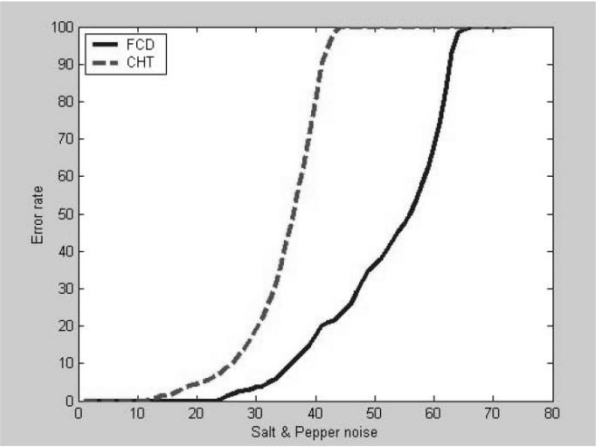


Abbildung 14: Beständigkeit der FCD und CHT gegenüber Pfeffer-und-Salz-Rauschen

2.2.3 Kantenbasierter Ansatz

Die Kantendetektion ist ein häufiger Ansatz, um mit einer geeigneten Weiterverarbeitung bestimmte Formen in einem Bild zu lokalisieren.

Verfahren zur Verstärkung von Kanten in einem Bild gibt es reichlich. Zu den häufig angewandten gehören bspw. der Sobel- und Canny-Operator [6, S. 120 ff.]. Nach einer Kantenverstärkung durch einen solchen Filter steht meist eine binäre Entscheidung an, welche Kantenpixel wirklich als Kantenpunkte betrachtet werden und welche nicht. Dies kann im einfachsten Fall durch eine Schwellwertoperation vollzogen werden. Das Ergebnis stellt ein Kantenbild dar [6, S. 129].

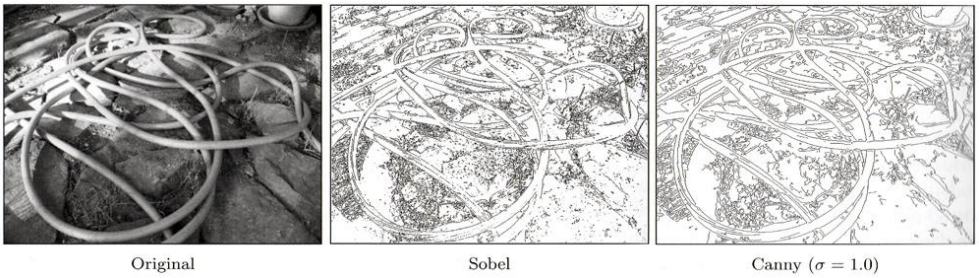


Abbildung 15: Ergebnisse von Kantendetektoren

Da ein reines Kantenbild selten perfekte Konturen zeigt, ist es zur Detektion von Kreisformen meist nicht ausreichend. Dennoch kann es als Grundlage dienen und mit einer sinnvollen Weiterverarbeitung zu dem gewünschten Detektionsergebnis führen.

Die indischen Entwickler P. Gupta, H. Mehrotra, A. Rattani, A. Chatterjee und A. K. Kaushik haben ein Verfahren zur Irisdetektion entwickelt, das mit Hilfe von Berechnungen über eine Kantendetektion erfolgt (Abb. 16) [12]. Aus Testbildern wurde eine Präzision der Detektion von 95,4% ermittelt [12, S.4].

Abbildung 16: Schritte der kantenbasierten Kreisdetektion am Beispiel einer Iris

3 Vorgehensweise der Markerdetektion

3.1 Bildvorverarbeitung

Da es sich bei den zu detektierenden Markern um solche mit bekannter Farbe handelt, ist es sinnvoll den Verarbeitungsprozess mit einer Segmentierung anhand der RGB-Werte zu beginnen.

Der erste Schritt ist, das Bild pixelweise zu durchlaufen und anhand eines Schwellenwerts zu entscheiden, ob das aktuell behandelte Pixel in einem neuen Binärbild den Wert 255 für Weiß oder 0 für Schwarz erhalten soll. In Hinsicht auf eine etwas geeignetere Darstellung bei der Ausgabe wird dabei die Markerfarbe im Binärbild Schwarz. Das RGB-Modell ist für diese Untersuchung geeigneter als das HSV-Modell. Eine Betrachtung der Mischverhältnisse von R, G und B unabhängig der Helligkeit ist einfacher in der Handhabung der Werte und anschaulicher. Die Kriterien für die rote Farbe (r) der Marker sind bezüglich Grün (g) und Blau (b) folgende:

$$r > 0$$
$$r \geq 2 * g$$
$$r \geq 2 * b$$

Eine deutlich von der gesuchten Farbe abweichende wird somit ausgeschlossen. Zugleich wird jeder Farbton, der dem Marker ähnlich ist, für den weiteren Prozess berücksichtigt.

Das nach diesem Schritt optimale Ergebnis des Maskenbildes würde den bzw. die Marker in Schwarz vor weitgehend weißem Hintergrund darstellen. Sollten weitere rote Objekte in der aufgenommen Szene sichtbar sein, so werden diese allerdings auch schwarz.

Ein perfektes Maskenbild wird in der Regel nicht erreicht, da zu viele Faktoren Einfluss nehmen, um dies zu verhindern. Ein häufiger Fall sind beispielsweise Störungen, die durch Reflektionen entstehen. Seien diese auf dem Marker verursacht oder durch Störelemente wie Staubpartikel, die sich zwischen dem Marker und der aufnehmenden Optik befinden. Das Resultat zeigt sich in dem Maskenbild. Es würden dort vereinzelt weiße Pixel entstehen, die sich in dem Marker bemerkbar machen. Eine Maßnahme ist angebracht, die Störpixel entfernt ohne die weiteren Bildelemente merklich zu verändern. Über gute Eigenschaften in dieser Hinsicht verfügt ein Medianfilter mit einem kleinen Radius.

3.2 Erstellung des Distanzmaßbildes

Ausgehend von dem binären Maskenbild der Markerfarbe wird das euklidische Distanzmaß berechnet. Das primäre Ziel hierbei ist es die Mittelpunkte der kreisförmigen Marker kenntlich zu machen und einen möglichen Radius der Marker aufzuzeigen.

Die Grundidee ist, dass das Distanzmaß dafür sorgt, dass sich der Mittelpunkt eines schwarzen Kreises vor weißem Hintergrund als ein

lokaler Minimalwert zu erkennen geben muss. Die Eigenschaft des Distanzmaßes, die Entfernung in Pixeln zum Rand eines Objekts für jeden Bildpunkt zu bestimmen, sorgt für das gewünschte Ergebnis.

3.3 Aufsuchen lokaler Minimalwerte

Ein Verfahren ist notwendig, das lokale Pixelwerte minimaler Helligkeit im Bild ermittelt. Für das weitere Vorgehen ist dies wichtig, damit der Radius vom Mittelpunkt zum Rand als Wert erfasst werden kann und anschließend eine Untersuchung, ob das jeweilige lokale Minimum von einem Marker umgeben ist, möglich ist.

Das Bestimmen lokaler Minimalwerte bedeutet, dass in einer begrenzten örtlichen Umgebung ein Pixel hinsichtlich seiner Helligkeit mit Umgebungspixeln verglichen wird. Das Bild wird hierbei pixelweise durchlaufen und pro Bildpunkt stehen sechs Pixelwertvergleiche an, die sich nach folgendem Prinzip ergeben:

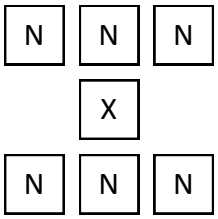
- Die erste Bedingung ist, dass nur Bildpunkte untersucht werden, deren Pixelwerte kleiner sind als ein Schwellenwert. Die Standardbelegung hierbei ist 240, was für einen Minimalradius r der Marker bedeutet:

$$r = pixelvalue_{max} - localThreshold = 255 - 240 = 15$$

Ist der Radius kleiner als 15, so wird das lokale Minimum nicht berücksichtigt.

Eine Schwelle ist sinnvoll um die Laufzeit der Berechnungen zu verkürzen und zu kleine Markerobjekte, die auch Bildstörungen sein könnten, von Beginn an auszuschließen.

- Ist der gerade zu behandelnde Pixelwert kleiner als die Schwelle, wird das aktuelle Pixel mit seinen Nachbarn verglichen:



X = aktuell behandeltes Pixel

N = Nachbarpixel

Randpixel des Bildes werden bei dieser Untersuchung bewusst vernachlässigt, da es dort keine Marker der Mindestgröße geben kann.

Sofern die Eigenschaft gegeben ist, dass X den gleichen oder einen kleineren Wert hat als alle seine N, ist ein lokales Minimum gefunden und wird hinsichtlich seiner x- und y-Koordinate gespeichert. Darüberhinaus wird auch verzeichnet, wie viele lokale Minima im Bild insgesamt gefunden wurden. Dies kann der Kontrolle dienen und hilfreich sein, um die Komplexität des Bildes zu erfassen.

Die Berechnung der lokalen Minima ist aufgrund der vielen Vergleichsoperationen relativ rechenaufwändig, allerdings werden so die möglichen Markermittelpunkte effektiv eingeschränkt. Zwar ergeben sich

auch lokale Minima, die nicht zu einem Marker gehören, dennoch wird die Anzahl der lokalen Minima nur noch einen Bruchteil an Werten hinsichtlich der gesamten Pixelanzahl des Bildes umfassen.

3.4 Verfahren zur Bestimmung von Markern

Eine Entscheidung, ob die lokalen Werte minimaler Helligkeit den Mittelpunkt eines Markers darstellen, muss in der Folge getroffen werden. Der Ansatz dabei ist, sich den zu ermittelnden Radius des Minimums zum Rand zu Nutze zu machen und an vermeintlichen Marker-Randpunkten im Bild zu prüfen, ob eine Kreisform vorliegt.

Der Radius lässt sich durch die einfache Berechnung *255 – Distanzmaß*, wobei das Distanzmaß den Pixelwert des lokalen Minimums erhält, bestimmen. Alternativ könnte man von dem Wert des Minimums pixelweise nach außen wandern und dabei die Schritte zählen, bis der Maximalwert 255, also der Objektrand, erreicht ist.

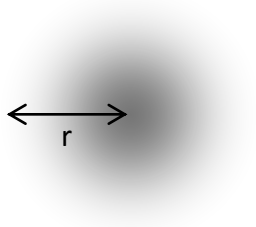


Abbildung 17: Radius r eines Markers im Distanzmaßbild

Durch ein anschließendes Prüfen an den Randpunkten des potentiellen Markerobjekts, kann erkannt werden, ob ein Marker vorliegt. Wenn der Wert an allen Pixeln der Objektkante (Abb. 18) sich über einem festgelegten Schwellwert bewegt, ist eine Kreisform gegeben.

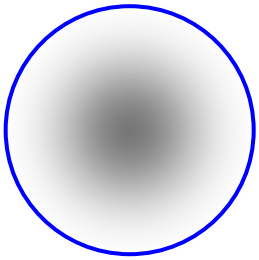


Abbildung 18: Auf ihre Pixelwerte zu untersuchende Punkte (blau) im Distanzmaßbild

Diese Überprüfung der Form garantiert, dass alle Objekte, die keine Kreisform haben, als potentielle Markerobjekte ausgeschlossen werden. Ein geeigneter Schwellwert mit einer nicht zu hohen Toleranz ist dabei vorausgesetzt.

4 Implementierung

4.1 Systemvoraussetzungen

Das Programm zur Marker-Detektion setzt ein installiertes Java Runtime Environment oder Development Kit voraus, wie es von Sun Microsystems [1] für alle gängigen Plattformen kostenfrei verfügbar ist. *MarkerDetection_.java* ist in der Programmiersprache Java geschrieben und als Plugin für die Bildverarbeitungssoftware ImageJ [2] angepasst. Auch ImageJ ist gemeinfrei[3] und ist somit für kommerzielle und nicht kommerzielle Zwecke kostenlos zu beziehen.

Entwickelt und getestet wurde das Programm mit folgenden technischen Hilfsmitteln:

Hardware

PC-System	Samsung P28 Notebook mit 15,4“-Bildschirm Intel Pentium M 1,60 GHz-Prozessor 1,00 GB Arbeitsspeicher
Kamerasystem	Nikon D200 Digitale Spiegelreflexkamera Sigma 10 – 20mm f/4 – 5.6 DC EX HSM Sigma 50mm Objektiv f/2.8 DG Nikon 18 – 135mm Objektiv f/3.5 – 5.6 IF-ED DX Sigma EF-500 DG SUPER Blitzgerät

[3] ImageJ unterliegt wie auch Java und Eclipse der Lizenz der freien Software

Software

Betriebssystem	Windows XP Service Pack 3
Java	Java Development Kit (JDK) 6 Update 5 ▪ Laufzeitumgebung und Java-Entwicklungstools; Erlaubt das Programmieren in Java
ImageJ	ImageJ 1.38x ▪ Plattformunabhängige Bildverarbeitungssoftware
Eclipse	Eclipse SDK 3.3.2 ▪ Entwicklungsumgebung für die Programmiersprache Java
Photoshop	Adobe Photoshop CS3 ▪ Professionelle Bildbearbeitungssofware

Für eine Ausführung des Programms muss der Ordner *MarkerDetection* mitsamt der Datei *MarkerDetection_.java* in den *plugins*-Ordner des ImageJ-Installationsordners kopiert werden. Wird ImageJ gestartet ist zuerst ein auf Marker zu untersuchendes Bild zu öffnen. Anschließend lässt sich das Plugin über die Menüleiste des Bildverarbeitungsprogramms mit der Befehlsabfolge *Plugins > MarkerDetection > MarkerDetection* starten.

Die Aufgabe des Programms ist es ein digital vorliegendes RGB-Bild auf bestimmte Markerobjekte zu untersuchen. Diese sind durch rote Schaumstoffbälle mit einem Durchmesser von etwa 8 cm repräsentiert. Das Schaumstoffmaterial hat die Eigenschaft auffällige Lichtreflexionen auf der Oberfläche des Balls weitgehend gut zu unterdrücken. Eine aufwändige programmiertechnische Berücksichtigung von stark gerichteten Reflexionen, wie sie bei glänzenden Oberflächen entstehen, ist so nicht gefordert.

Ziel der Ausgabe ist es, die richtige Anzahl an vorhandenen Markern in der Szene mit den jeweiligen Werten der Position und Größe im Bild zu detektieren. Nach der einzelnen Anzeige der gefundenen Marker wird das Resultat in einer Tabelle dargestellt, in der alle Ergebniswerte zusammengefasst sind. Eine eventuelle Methode zur Weiterverarbeitung könnte dann auf diese Tabellenwerte zurückgreifen.

4.2 Aufbau des Plugins

Das Plugin zur Detektion von Markern besteht aus der Datei *MarkerDetection_.java*. Die Datei gliedert sich in Importe bereits vorhandener Pakete und Klassen, auf die in der anschließenden Klasse der *MarkerDetection_* zurückgegriffen wird. Die Importe sind im Einzelnen:

```
import ij.*;
import ij.process.*;
import ij.plugin.filter.RankFilters;
import ij.plugin.filter.PlugInFilter;
import ij.measure.ResultsTable;
```

Notwendig ist dies, um auf bereits implementierte Verfahren (wie z. B. die Berechnung des Distanzmaßes oder die Ausgabe von Ergebnisfenstern) zurückgreifen zu können.

Die Klasse *MarkerDetection_* beginnt mit der Deklaration globaler Variablen, die für den Austausch berechneter Werte über Methoden innerhalb der Klasse notwendig sind. Diese Variablen sind durch den Zugriffsbezeichner *private* gekapselt, damit eine Datenmanipulation dieser Werte nicht unberechtigt über andere Klassen erfolgen kann [13, S. 387].

Die anschließende *setup*-Methode gibt in erster Linie an, dass es sich bei dem zu prozessierenden Originalbild um ein RGB-Bild handelt (*DOES_RGB*), die folgende *run*-Methode strukturiert den Ablauf der Methoden. Beim Starten des Programms wird diese Methode aufgerufen, die wiederum alle weiteren in der Klasse befindlichen Methoden nach sequentieller Abfolge

aufruft. Die mathematischen Berechnungen und Objekterzeugungen erfolgen dabei weitgehend in den einzelnen Methoden.

Noch vor dem Schritt der Segmentierung ist wichtig, dass zuerst die RGB-Werte des Bildes getrennt und in positive Werte gewandelt werden. Dies erfolgt durch ein bitweises *UND* mit der Maske *0xff* und ein verschieben der Werte in der Methode *getMarkerPixel* [6]:

```
int r, g, b;
r = ((i >> 16)&0xff);
g = ((i >> 8)&0xff);
b = (i & 0xff);
```

Danach setzt die Segmentierung an und erstellt über das neue *ImagePlus*-Objekt *newMask* ein neues Binärbild anhand der Bedingung für die Markerfarbe. Es entstehen schwarze, potentielle Markerbereiche vor weißem Hintergrund. Anschließend wird ein Medianfilter angewandt, der über die ImageJ-Klasse *RankFilters* implementiert ist. Der Radius ist über die globale Variable *medianRadius* mit dem Wert 1 voreingestellt.

Der nächste Schritt ist die Erstellung des Distanzmaßes. Das Verfahren hierzu wird direkt von ImageJ übernommen und auf das Binärbild angewandt:

```
IJ.run("Distance Map");
```

Auf das Distanzmaßbild wird nun die Methode *getLocalMin* ausgeführt. Im Zentrum steht hier das Verfahren lokale Werte minimaler Helligkeit aufzufinden, was über Pixelvergleiche in zwei verschachtelten *for*-Schleifen geschieht. Um die Vergleiche nicht unnötig auf alle Pixel anwenden zu

müssen, ist mit dem Schwellenwert *localThreshold* ein Maximalwert angegeben. Nur Pixel mit einem Wert, der unter dem global festgelegten *localThreshold* von 240 liegen, werden in der inneren Schleife berücksichtigt. Die Ergebnisse zu diesem Zeitpunkt sind alle lokalen Minima *n* des Distanzmaßbildes in ihrer gesamten Anzahl und mit ihren einzelnen Koordinaten:

```
xCoord[n];
yCoord[n];
numberOfLocalMin;
```

Die Methode *getMarkers* sorgt für den Ausschluss lokaler Minimalwerte und das Detektieren tatsächlicher Marker. Zuerst wird der Radius der lokalen Minima bestimmt, indem von dem Minimalwert die Pixelanzahl nach links außen, bis der Wert 255 erreicht wird, gezählt wird. Dieses Verfahren verlangt mehr Rechenzeit als die bloße Berechnung von *255 – Distanzmaß*, ergab sich aber in der Praxis bei vielen Testbildern als visuell genauer im Endergebnis. Die Ursache hierfür liegt offenbar an dem Mangel einer perfekten Kreisform im Originalbild, einhergehend mit mehreren lokalen Minimalwerten für einen Markermittelpunkt. Möglicherweise kompensieren sich an dieser Stelle die Ungenauigkeiten bei dem Aufsuchen des Mittelpunktes und die Ungenauigkeit bei der iterativen Bestimmung des Radius mit nur einer Vorzugsrichtung. In der Theorie sollte die erwähnte Berechnung von *255 – Distanzmaß* ein exakteres Resultat liefern, das Endergebnis des Markers und seine Visualisierung in der letztendlichen Ausgabe ist mit dem aufwändigeren Verfahren in der Praxis allerdings in vielen Fällen zentrierter.

Bevor nun der eigentliche Kern der Detektion folgt ist noch ein Schritt sinnvoll um den Rechenaufwand zu verringern. Die boolesche Variable *possibleMarker* erhält dabei nur ein *true* für einen möglichen Marker, wenn der zugehörige Radius größer als 5 ist. Da das Verfahren der Auffindung lokaler Minima die direkten Nachbarn links und rechts des prozessierten Pixels nicht vergleicht, können bspw. lokale Minima mit einem Radius von 0 oder 1 auftreten. Damit diese viel zu kleinen Minima nicht unnötig weiter behandelt werden, sollen sie ausgeschlossen werden.

Das Prinzip der Detektion einer Kreisform erfolgt in verschachtelten for-Schleifen und Abfragen von Bedingungen. Durch den Radius und das Prüfen an der Kreiskante wird mit einer Toleranz von *threshold = 240* festgelegt, ob ein Kreis das lokale Minimum aus *possibleMarker* umgibt.

Ausgangspunkt dabei ist der Mittelpunkt M mit seinen x- und y-Koordinaten. Durch den gegebenen Radius r lassen sich alle vermeintlichen Randpunkte der Objektform lokalisieren (Abb. 19). Im ersten Schritt werden im Wechsel alle Pixelwerte von P1 nach P2 und P3 nach P4 geprüft.

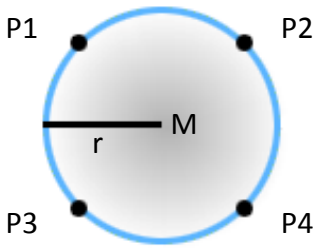


Abbildung 19: Parameter eines lokalen Minimums im Distanzmaßbild mit Kennzeichnung möglicher Randpunkte (blauer Kreis)

Sobald sich ein Pixelwert an einer Stelle außerhalb des Toleranzbereiches bewegt, also kleiner als 240 ist, vollzieht sich ein *break* und der Schleifendurchlauf wird abgebrochen. Eine Kreisform liegt dann nicht vor, der *possibleMarker* wird auf *false* gesetzt und es wird der nächste Markerkandidat behandelt. Sind aber alle Punkte von P1 bis P2 und P3 bis P4 tatsächliche Randpunkte, vollzieht sich die Prozedur der Pixelwertbetrachtung auf gleiche Weise auch für alle Punkte von P1 nach P3 und P2 nach P4. Erst wenn auch hier alle Punkte als Markerrandpunkte verifiziert wurden, ist ein Marker gefunden und der Wert *true* in possibleMarker bleibt bestehen.

Nach diesen Berechnungen wird das Array *possibleMarker* in die globale Variable *marker* geschrieben. Die eindeutigen Bezeichnungen sollen dabei der Übersichtlichkeit dienen: Das Detektionsverfahren erhält Werte über *possibleMarker* und gibt letztendlich alle gefundenen Objekte in *marker* aus.

Da es bei einem Distanzmaßbild bei nicht-idealen Kreisformen vorkommen kann, dass lokale Minimalwerte gefunden werden, die in unmittelbarer Nähe voneinander liegen, muss dies berücksichtigt werden. Es kann dadurch nämlich der Fall eintreten, dass Marker gefunden werden, die sich in ihrem Mittelpunkt und Radius nur um wenige Pixel unterscheiden. Um solche Begebenheiten auszuschließen erfolgt eine Bedingung, dass es in dem Detektionsergebnis keine sich überschneidenen Marker geben kann. Rechnerisch darf demnach die Distanz von den Mittelpunkten zweier Marker nicht kleiner sein als die Summe ihrer Radien. Die Variable *marker*

wird dabei schrittweise durchlaufen und vergleicht die Parameter aufeinanderfolgender Marker. Gegebenenfalls erhält hierbei ein Marker noch ein *false*.

Letztendlich verfügt *marker* mit seinen *true* oder *false*-Angaben über das Endergebnis der Detektion. Mit der Bedingung, dass *n < numberOfLocalMin* sein muss, lassen sich alle Ergebniswerte über folgenden Variablen auslesen:

```
numberOfLocalMin;
marker[n];
radius[n];
xCoord[n];
yCoord[n];
```

4.3 Ausgabe der Ergebnisse

Am Ende der Klasse *MarkerDetection_* erfolgt die Ausgabe der ermittelten Resultate. Es wird für jeden Eintrag des booleschen Arrays geprüft, ob ein Marker vorliegt. Ist dies der Fall werden die weiteren zugehörigen Daten ausgelesen und über die ImageJ-Methode *makeOval* ein Kreis an die entsprechende Stelle gezeichnet, wo sich der Marker befindet (Abb. 20).

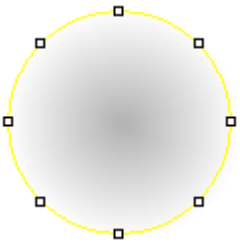


Abbildung 20: Visualisierung eines aktuell gefundenen Markers im Distanzmaßbild

Darüberhinaus wird über *IJ.showMessage* ein Dialogfenster mit den Werten des aktuell sichtbaren Markers ausgegeben (Abb. 21).

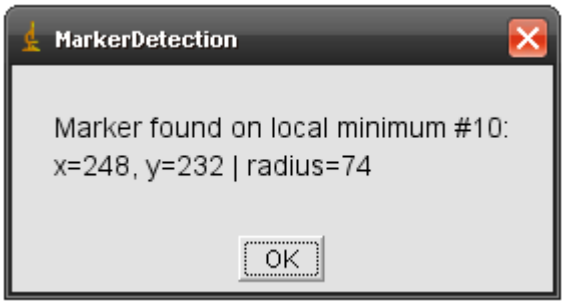


Abbildung 21: Ergebnisfenster eines aktuell gefunden Markers

Diese angezeigten Werte werden anschließend noch in zuvor initialisierten Variablen geschrieben, die zu der ImageJ-Klasse *ResultsTable* gehören. Anschließend wiederholen sich diese Schritte für den nächsten gefundenen Marker.

Sind alle Marker visuell kenntlich gemacht und mit ihren Werten dargestellt worden, erfolgt der letzte Schritt der Ergebnisausgabe. Die

Erstellung eines Objekts des Typs *ResultsTable* zu Beginn der Ausgabemethode, das Verwenden zugehöriger Methoden und das Belegen von Variablen mit Ergebniswerten kommt nun zum Tragen. Es wird eine Tabelle mit allen wichtigen Werten zu jedem einzelnen Marker angezeigt (Abb. 22).

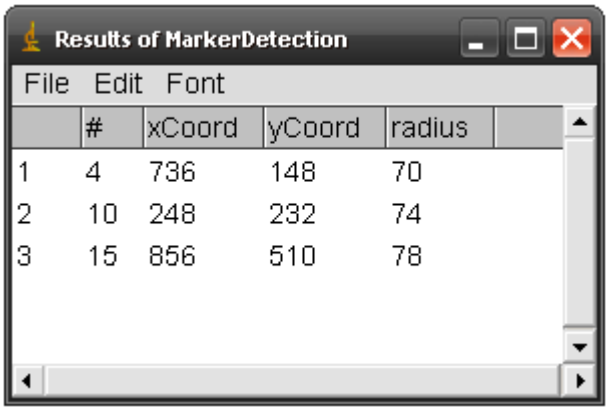


Abbildung 22: Beispiel für eine Ergebnistabelle aller gefundenen Marker

Für den Fall, dass im gesamten Bild kein einziger Marker gefunden wurde, wird dieses Ergebnis wie folgt quittiert (Abb. 23):

Abbildung 23: Ausgabe, wenn kein Marker gefunden wurde

5 Ergebnisse

5.1 Markerbilder

Nachfolgend sind beispielhafte Resultate von Bildern mit Marker aufgeführt, bei denen die Detektion zu einem korrekten Ergebnis führt.

Das Markerbild 1 (Abb. 24) zeigt die Erkennung der gesuchten Objekte an einem einfachen Beispiel. Die Marker grenzen sich deutlich von ihrer Umgebung ab, was für eine korrekte Detektion sorgt (Abb. 21).

Markerbild 2 (Abb. 26) ergibt ein komplexeres Distanzmaßbild (Abb. 27). Die kreisförmigen Objekte, die keine Marker sind, werden durch ihre von Rot abweichenden Farbelemente bei der Segmentierung gestört und sind im Distanzmaßbild eher schwach dargestellt. Da es sich dort, wie auch bei den beiden Quadraten, nicht um eine Kreisform handelt, werden alle Objekte bis auf die tatsächlichen Marker ausgeschlossen und nicht detektiert.

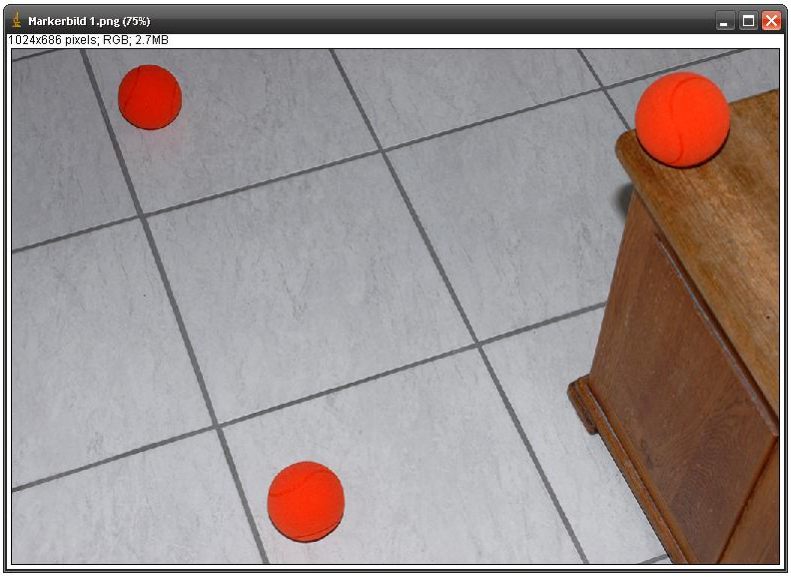


Abbildung 24: Markerbild 1

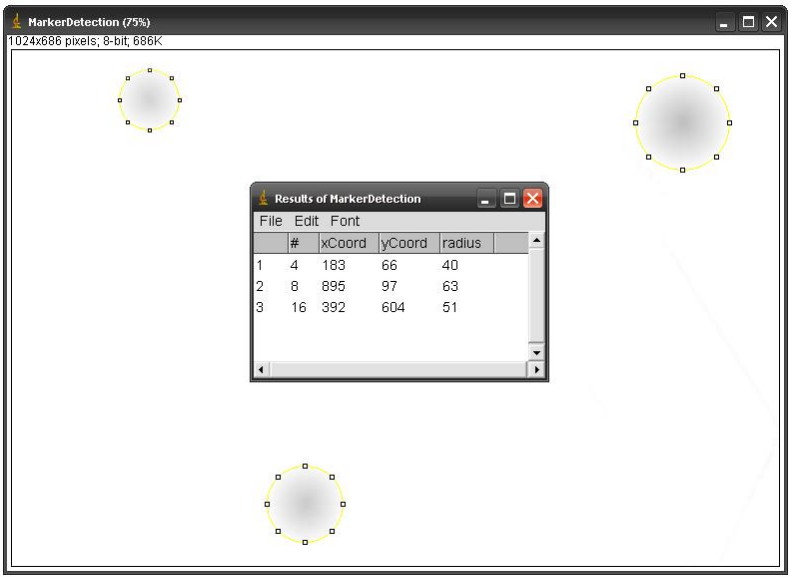


Abbildung 25: Detektionsergebnis von Markerbild 1

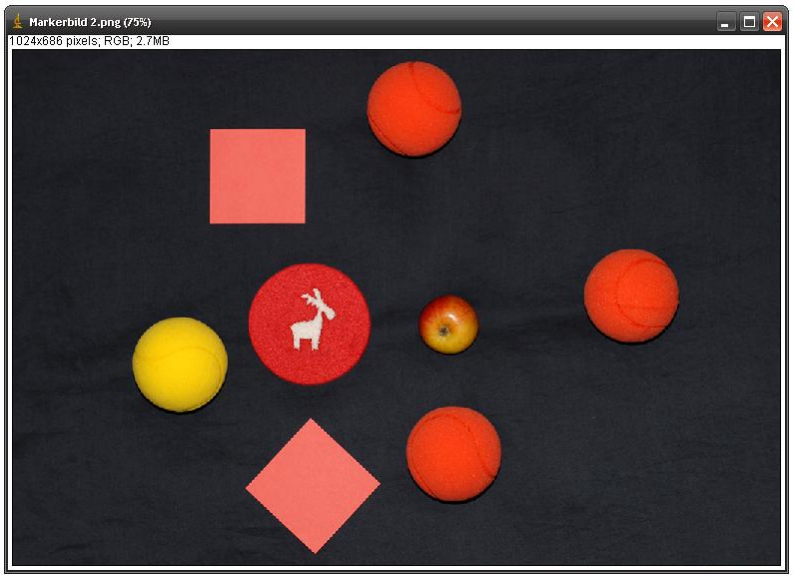


Abbildung 26: Markerbild 2

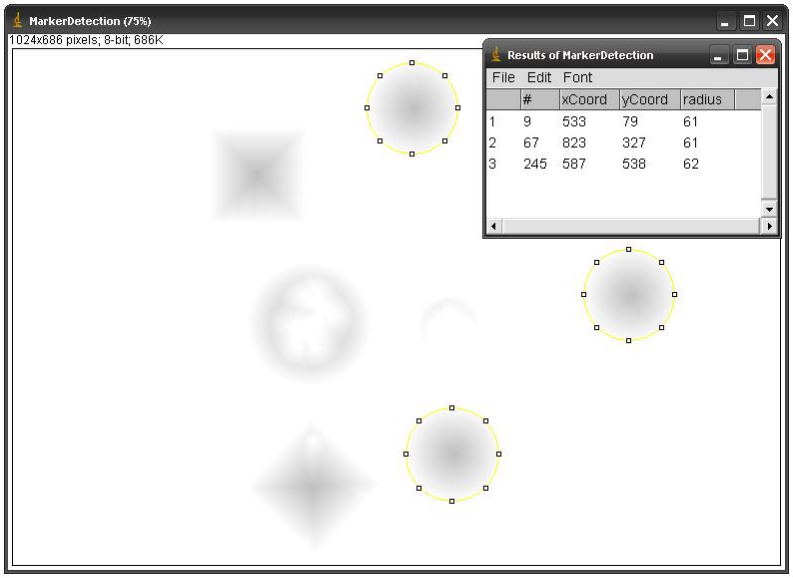


Abbildung 27: Detektionsergebnis von Markerbild 2

5.2 Grenzsituationen der Markererkennung

Unter bestimmten Bedingungen stößt die Markerdetektion an ihre Grenzen. Die Ursachen lassen sich in verschiedene Kategorien einordnen.

5.2.1 Farbveränderung

Eine starke Veränderung der Markerfarbe führt bereits bei der Segmentierung zu Problemen. Die Farbtoleranz in dem Programm ist allerdings bewusst sehr groß gehalten. Handelt es sich aber um einen deutlich von Rot abweichenden Farbton, werden die Marker (außer bei einer Anpassung des Programmcodes) nicht mit in das Binärbild übernommen. Eine solche Situation kann durch einen falschen Weißabgleich der aufnehmenden Kamera entstehen.

Das folgende Bild wurde im Rohdatenformat aufgenommen und bei der Entwicklung beabsichtigt mit einem völlig ungeeigneten Weißabgleich versehen (Abb. 28). In dieser Extremsituation ist eine Markerdetektion nicht möglich (Abb. 29).

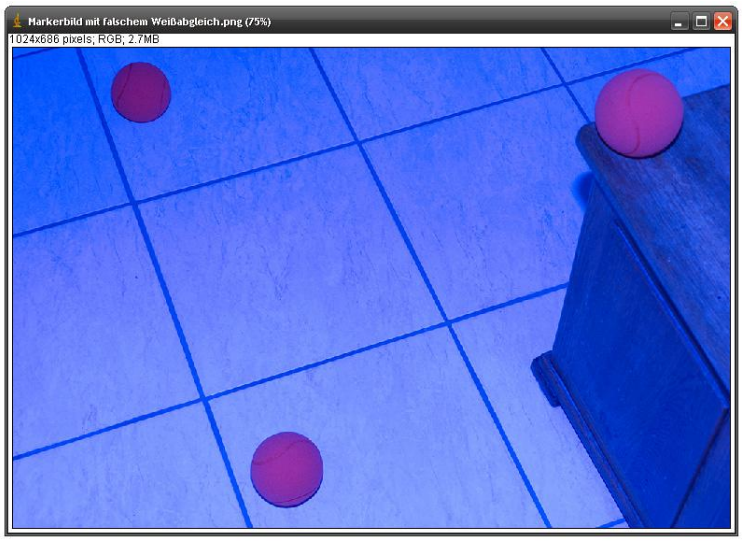


Abbildung 28: Markerbild mit einem künstlich erstellten, falschen Weißabgleich

Abbildung 29: Detektionsergebnis bei falschem Weißabgleich

5.2.2 Formveränderung

Ist ein Marker in einem Bild zu klein, wird er nicht berücksichtigt. Der Grund ist die Schwierigkeit der Unterscheidung zwischen Bildstörungen und Markerobjekten. Der Implementierung nach erfolgt eine Detektion sobald ein Marker einen Radius von mindestens 15 Pixeln im Bild hat. Testergebnisse belegen dieses Verhalten.

Ein Marker darf allerdings auch nicht zu groß sein, da sich sonst das Distanzmaßbild nicht für eine Detektion lokaler Minima eignet. Das Distanzmaß bestimmt für jeden Pixel den Abstand zu der nächsten Kante, wozu allerdings nur 256 Werte zur Verfügung stehen. Somit kann, auf Kreisformen bezogen, ein Markerradius von 255 mit einer 0 als Wert im Zentrum gerade noch dargestellt werden. Liegt ein Radius bei bspw. 400, ist der Wert 0 vom Rand aus betrachtet vor dem Kreismittelpunkt erreicht. Alle folgenden Pixel in Richtung des Kreiszentrums werden dann ebenfalls zu 0, womit sich ein komplett schwarzer Kreis mit dem Radius 400 – 255 = 145 innerhalb der Markerform ergibt. Bei einem sehr großen Bild mit Markerradien von über 255 (Abb. 30) können somit keine Marker detektiert werden (Abb. 31, oberer und linker Marker). Es gibt kein zentrales, lokales Minimum eines Markers, das durch ein einziges Pixel dargestellt wird, was für das Programm bei der Suche nach lokalen Minima notwendig ist.

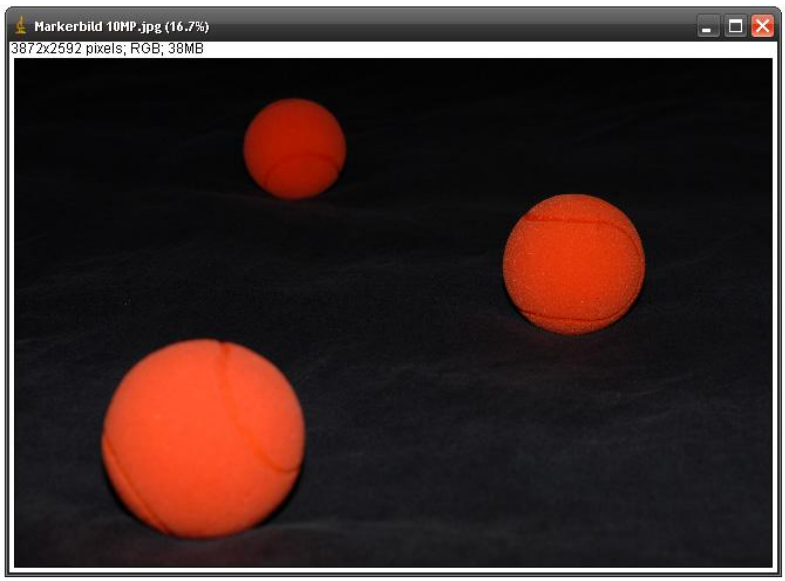


Abbildung 30: Großes Markerbild (3872 x 2592 px)

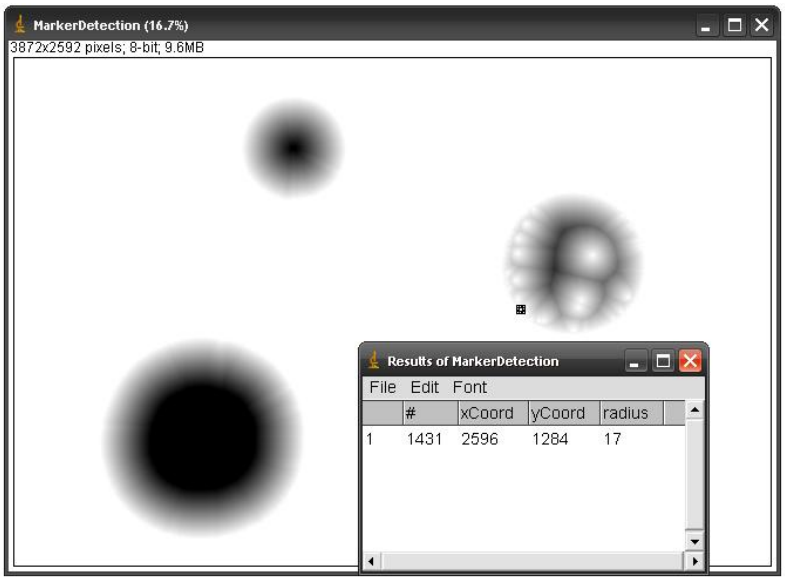


Abbildung 31: Ergebnis der Markerdetektion bei zu großen Markern

Ist ein Bild mit großen Markern zu untersuchen, kann sich ein weiteres Problem ergeben. Je größer das Bild, desto größer sind auch Bildstörungen, die nach der Segmentierung für ein unsauberes Binär- und Distanzmaßbild sorgen. Diese Störungen würden den weiteren Prozess beeinflussen und unter Umständen zu einer Fehlinterpretation eines Markers führen (Abb. 31, rechter Marker). Hier könnte allerdings ein zuvor größer gewählter Radius des Medianfilters diesem Problem vorbeugen.

Ist die runde Form des Markers beeinträchtigt, wird die Detektion erschwert, da die Rundheit ein Hauptkriterium der Detektion darstellt. Eine Formveränderung kann bspw. während der Aufnahme erfolgen, wenn die Kamera über ein Objektiv mit einer extrem kurzen Brennweite verfügt. Sind im normalen Weitwinkel- bis hin zum Telebereich keine großen Bildverzerrungen zu erwarten (Abb. 32.), so sind im Ultraweitwinkelbereich[4] Kreisformen die nicht im Bildzentrum liegen nicht mehr als solche zu erkennen (Abb. 34, 35). Dieses Beispiel bezieht sich allerdings auf eine praktische Anwendung von unkalibrierten Objektiven. Bei einer idealen Lochkamera wie auch bei entsprechend kalibrierten Kameras treten solche Verzerrungen theoretisch nicht auf.

[4] Bilder, die mit einem Objektiv mit einem diagonalen Bildwinkel von über 80° erzeugt werden (meist bei einer kleinbildäquivalenten Brennweite von 24mm oder weniger)

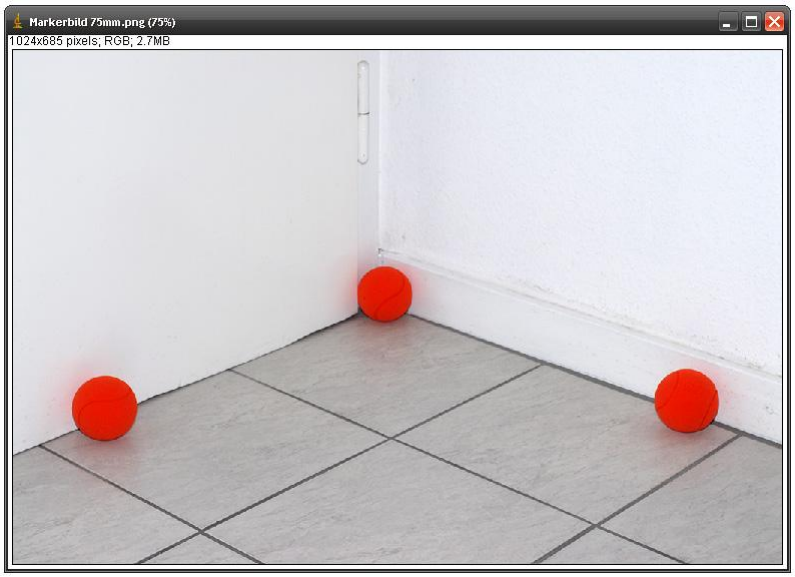


Abbildung 32: Markerbild bei einer kleinbildäquivalenten Brennweite von 75mm

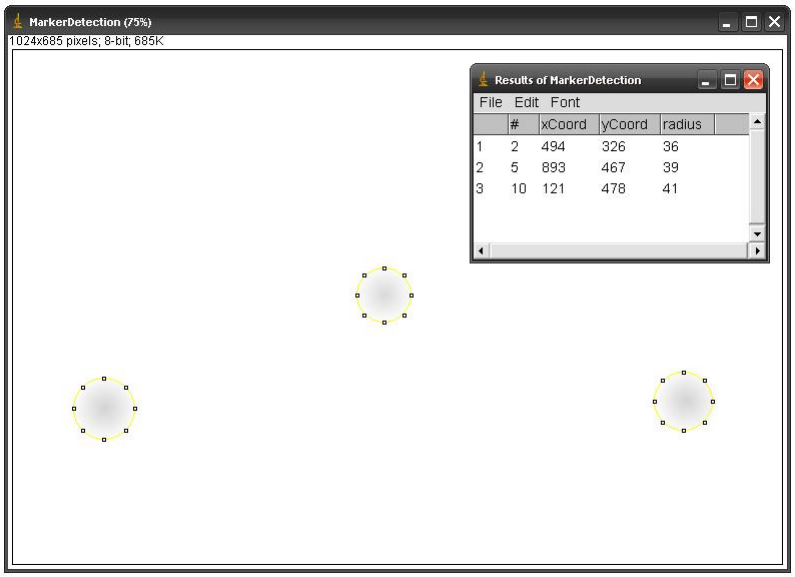


Abbildung 33: Ergebnis der Markerdetektion (75mm-Bild)

Abbildung 34: Markerbild bei einer kleinbildäquivalenten Brennweite von 15mm

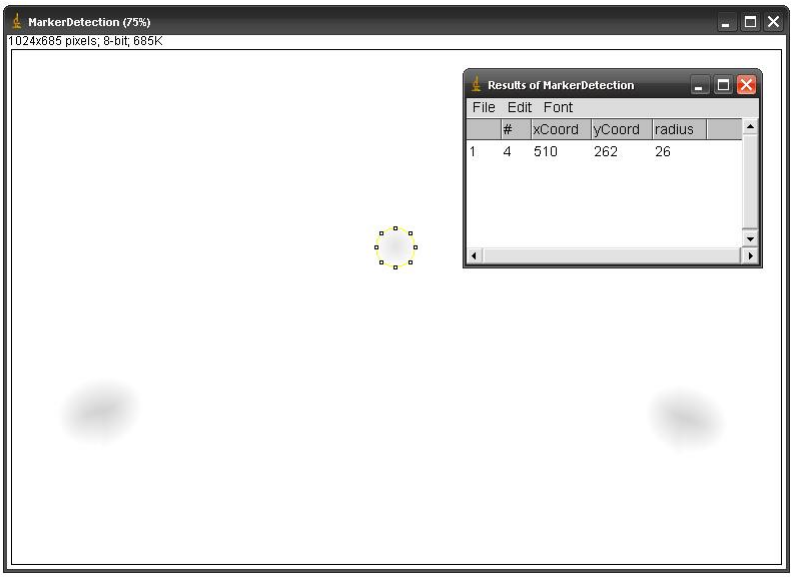


Abbildung 35: Ergebnis der Markerdetektion (15mm-Bild)

Bei einer sehr kurzen Brennweite nehmen die Marker eine stärkere Ellipsenform an, je näher sie sich am Bildrand befinden. Eine Kreisdetektion im Distanzmaßbild ist dabei nur bei dem weitgehend unverzerrten Marker in der Mitte des Bildes möglich.

5.2.2.1 Beleuchtungsänderung

Eine Änderung der Beleuchtung kann in Grenzsituationen eine visuelle Formveränderung von Objekten bewirken. Mit einer seitlichen oder von oben gerichteten Lichtquelle können Abschattungen an den Markern entstehen, die eine Detektion der roten Farbe in gewissen Bereichen nicht mehr zulassen. Bereits bei der Segmentierung würden keine Kreisformen erkannt.

5.2.2.2 Überlappung

Eine Überlappung von Markern führt bildtechnisch gesehen auch zu einer Formveränderung. Ist die Szene homogen ausgeleuchtet, gibt es keine oder nur sehr geringe Schattenwürfe an den Markern, die bei der Farbsegmentierung für ein Verschmelzen von sich überdeckenden Markern sorgt. Nach der Segmentierung ist keine Kreisform mehr erkennbar, was sich entsprechend in das Distanzmaßbild fortsetzt (Abb. 37).

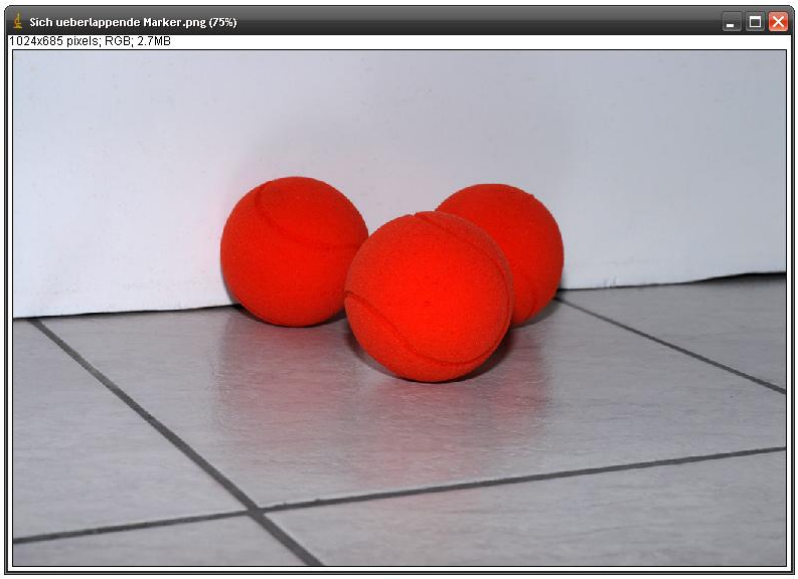


Abbildung 36: Sich überlappende Marker

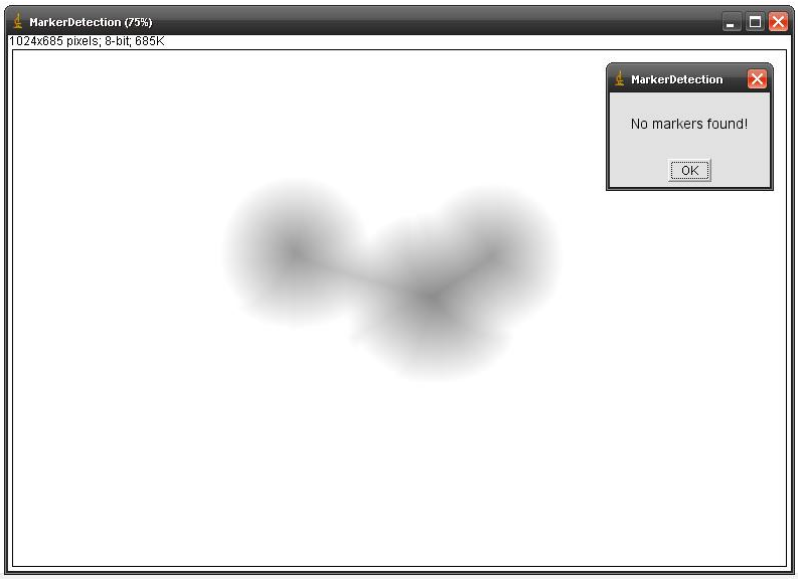


Abbildung 37: Distanzmaßbild und Detektionsergebnis bei einer Überlappung der Marker

6 Diskussion

Die Ergebnisse zeigen, dass das Programm ein effektives und weitgehend präzises Detektionsverfahren bei geeigneten Markerbildern darstellt. Die Probleme liegen in der Erkennung von großen Markerobjekten. Überschreitet der Radius eines Markers den Wert von 255 ist das Distanzmaßbild ungeeignet für eine Erkennung lokaler Minima. Eine programmiertechnische Anpassung wäre zwar denkbar (bspw. durch ein 16-Bit-Graustufenbild), ob es allerdings sinnvoll ist solch große Objekte erkennen zu wollen, sei dahingestellt. Eine Vorverarbeitung des zu untersuchenden Markerbildes (z. B. eine Verkleinerung von sehr großen Bildern) ist je nach Anwendungszweck gegebenenfalls angebracht.

Die ursprüngliche Idee einer Umsetzung des Detektionsverfahrens lag bei dem Ansatz einer Faltung des Originalbildes mit anschließender Betrachtung der Kanten. Durch diese rechenaufwändigen Verfahren wäre die Laufzeit allerdings sehr hoch gewesen. Da sich das Distanzmaßbild als eine gute Grundlage zur Detektion erwies, wurde das Hauptaugenmerk auf dieses Verfahren gelegt und letztendlich umgesetzt. Einer subjektiven Beurteilung nach werden die gesuchten Objekte durch das Programm ohne große Verzögerung erkannt. Ein Kritikpunkt dürfte im direkten Vergleich mit bspw. gradientenbasierten Detektionsverfahren dennoch der höhere Rechenaufwand sein.

Ein aufsetzendes Verfahren, dass die Ergebnisse der Markerdetektion weiter behandelt, wäre sinnvoll. Anwendungsgebiete gibt es reichlich und

die Möglichkeiten, die ermittelten Werte weiter zu verarbeiten, sind vielfältig.

7 Literaturverzeichnis und Quellenangaben

[1] Sun Microsystems: *Sun Developer Network: The Source for Java Developers*, Internetquelle: http://java.sun.com/ (Stand: 04.11.2008)

[2] Wayne Rasband: *ImageJ: Image Processing and Analysis in Java*; Internetquelle: http://rsbweb.nih.gov/ij/ (Stand: 04.11.2008)

[3] Eclipse Foundation: *Eclipse*; Internetquelle: http://www.eclipse.org/ (Stand 04.11.2008)

[4] A. A. Rad, K. Faez, N. Qaragozlou; *Fast Circle Detection Using Gradient Pair Vectors*; Proc. 7th International Conference on Digital Image Computing: Techniques and Applications (DICTA 03; Sydney, Australien); 2003; Internetquelle: http://www.cmis.csiro.au/hugues.talbot/dicta2003/cdrom/pdf/0879.pdf (Stand: 04.11.2008)

[5] W. K. Pratt: *Digital Image Processing*; Verlag Wiley-Interscience; 4. Auflage (2007)

[6] W. Burger, M. J. Burge: *Digitale Bildverarbeitung: Eine Einführung mit Java und ImageJ*; Springer-Verlag Berlin Heidelberg; 2. Auflage (2006)

[7] B. Jähne: *Digitale Bildverarbeitung*; Springer-Verlag Berlin Heidelberg; 6. Auflage (2005)

[8] U. Baufeldt, H. Rösner, J. Scheuermann, H. Walk: *Informationen übertragen und drucken*; Verlag Beruf + Schule; 14. Auflage (2000)

[9] H. Bässmann, J. Kreyss: *Bildverarbeitung Ad Oculus*; Springer-Verlag Berlin Heidelberg; 4. Auflage (2004)

[10] S. J. K. Pedersen: *Circle Hough Transform*; Aalborg University, Vision, Graphics and Interactive Systems; 2007; Internetquelle: http://www.cvmt.aau.dk/education/teaching/e07/MED3/IP/Simon_Pedersen_CircularHoughTransform.pdf (Stand: 04.11.2008)

[11] P. Gupta, H. Mehrotra, A. Rattani, A. Chatterjee, A. K. Kaushik: *Iris Recognition using Corner Detection*; Internetquelle: www.security.iitk.ac.in/contents/repository/papers/Corner.doc (Stand: 04.11.2008)

[12] Pentax; *Produktbeschreibung Pentax Optio S12: Objekterkennung innerhalb des Fokusrahmens*; Internetquelle: http://www.pentax.de/de/product/17025/body/overview/digitale_kompaktkameras.php (Stand: 04.11.2008)

[13] C. Ullenboom: *Java ist auch eine Insel: Programmieren mit der Java Standard Edition Version 6*; Verlag Galileo Computing; 7. Auflage (2008)

[14] A. Weber: *Farbatlas Pathogene Protozoen*; Bildatlas (1970)

8 Abbildungsverzeichnis

9 Anhang

Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne fremde Hilfe verfasst und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.

Köln, den 04.11.2008

Unterschrift
(Felix Spalthoff)

Sperrvermerk

Die vorgelegte Arbeit unterliegt keinem Sperrvermerk.

Weitergabeerklärung

Ich erkläre hiermit mein Einverständnis, dass das vorliegende Exemplar meiner Bachelorarbeit oder eine Kopie hiervon für wissenschaftliche Zwecke verwendet werden darf.

Köln, den 04.11.2008

Unterschrift
(Felix Spalthoff)